ADVANCED DATA MODEL PATTERNS

David C. Hay
Essential Strategies, Inc.

The book *Data Model Patterns: Conventions of Thought*[1] describes a set of standard data models that can be applied to standard business situations. These patterns, it turns out, occur on several levels. At the basic level are models of the things seen in business. The patterns in the book are a bit more abstract than conventionally seen, but they do describe things that are easily recognizable to anyone: people and organizations, products, contracts, and so forth.

There is a more abstract level of modeling, however, which is necessary when the things being modeled don't fall into these tidy categories. This level, also described in the book, is the subject of this paper.

THE BASIC MODEL

Before getting into the more exotic models, it is useful to be sure we understand the basic patterns that will apply to nearly all organizations. Each real organization will have variations on this model, but here you will find the elements that will be present in nearly every one. Figure 1, for example, shows that the entity PARTY encompasses PERSON and ORGANIZATION. That is, a PERSON and an ORGANIZATION are each things of significance, and if you want to refer to either, you can refer to a PARTY.

PARTIES may be related to each other, as shown by the entity PARTY RELATIONSHIP. This is simply the fact that one PARTY has a specified relationship with another, as in a reporting structure, employment, marriage, membership in a club, etc.

A PARTY may have more than one address. Each address is shown in this model as a SITE, where each PARTY may be *located via* one or more PARTY PLACEMENTS *in* a SITE. Each SITE must be *in* one or more GEOGRAPHIC AREAS, such as a city or region.

[1] David Hay, *Data Model Patterns: Conventions of Thought*, Dorset House Publishers, Inc. (New York: 1996). This article is largely derived from this book.

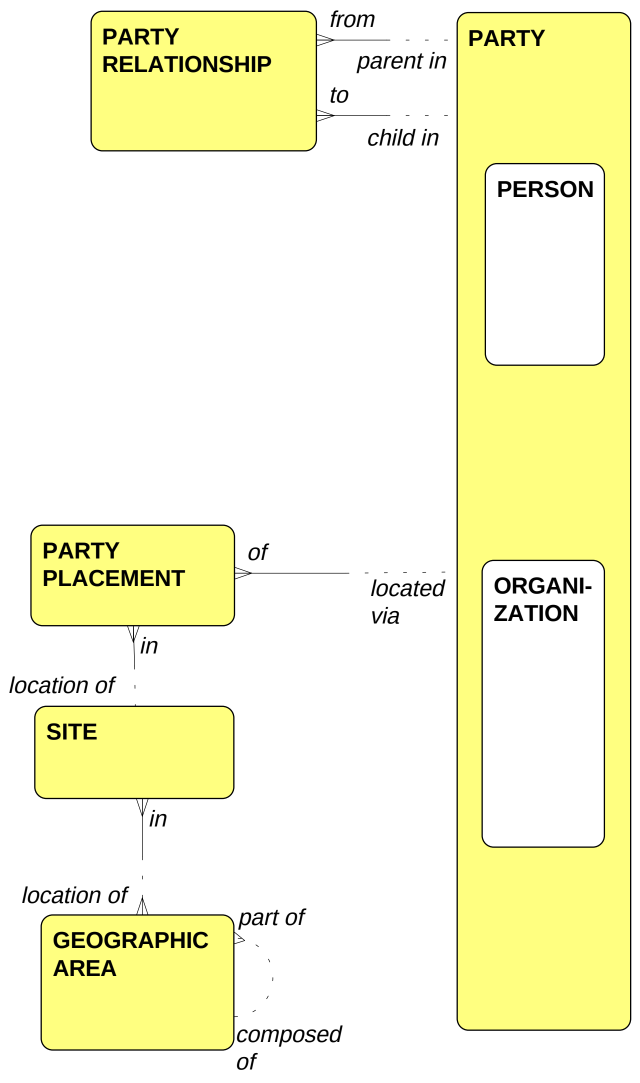

Figure 1: People and Organizations

Figure 2 shows the "stuff" a company deals with. Here it is called PRODUCT TYPE and PRODUCT INSTANCE. It could be called ASSET TYPE and ASSET, ITEM TYPE and ITEM OCCURRENCE, or something similar. Note the distinction between PRODUCT INSTANCE, a physical example of the product, and PRODUCT TYPE, which is the definition of it, such as you would see in a catalogue.

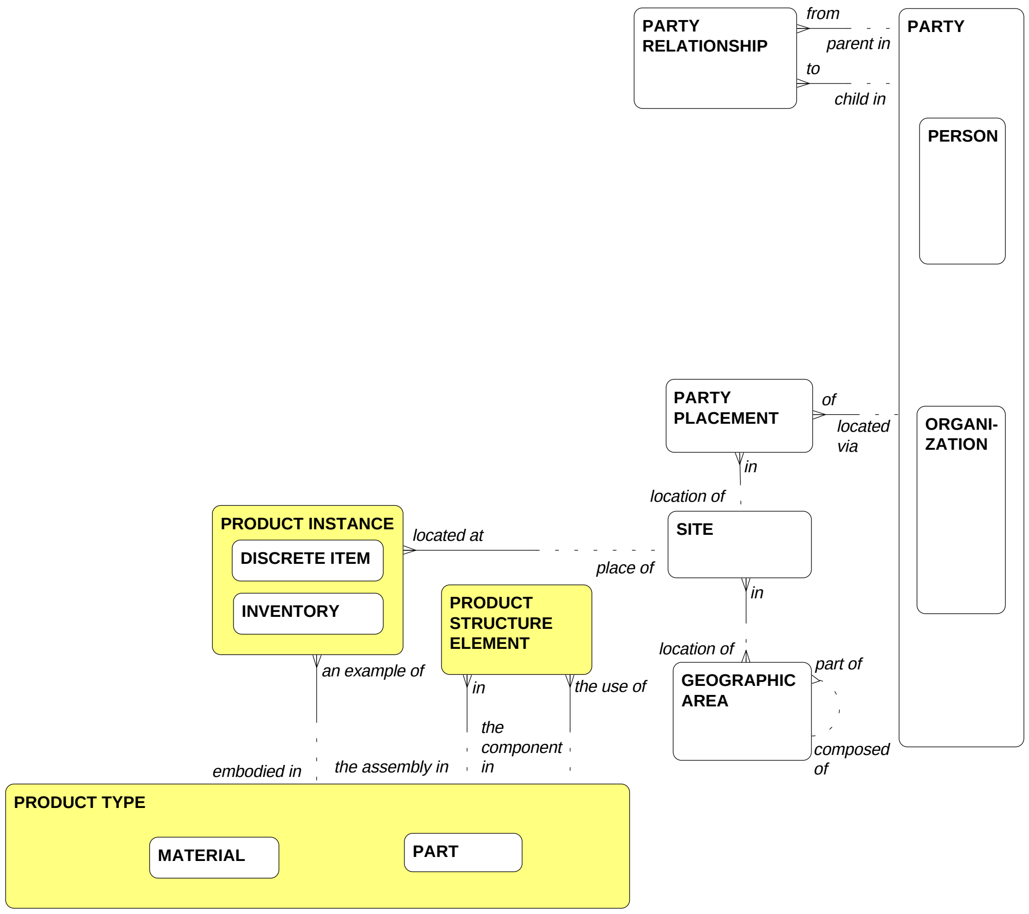

Figure 2: Product Types

A PRODUCT STRUCTURE ELEMENT is the fact that one PRODUCT TYPE may have another PRODUCT TYPE as a component. Thus an assembly may have three sub-assemblies as components, and this would be represented by three PRODUCT STRUCTURE ELEMENT occurrences where the assembly is *the assembly in* and each sub-assembly is *the component in* each PRODUCT STRUCTURE ELEMENT, respectively.

Note that a PRODUCT INSTANCE may be either a DISCRETE ITEM which is kept track of individually, or an INVENTORY which is a collection of items. In either case, each PRODUCT INSTANCE must be *at* a SITE.

Figure 3 shows AGREEMENT, where an AGREEMENT is any formal relationship between two PARTIES. Typically, this is a purchase order or a sales order, but it may encompass other kinds of AGREEMENTS as well. Invariably, our organization is one of the parties – either the *buyer in* the AGREEEMENT if it is a purchase order, or the *seller in* the AGREEMENT if it is a sales order.

Each AGREEMENT must be *composed of* one or more LINE ITEMS, where each LINE ITEM is *for* a PRODUCT TYPE.

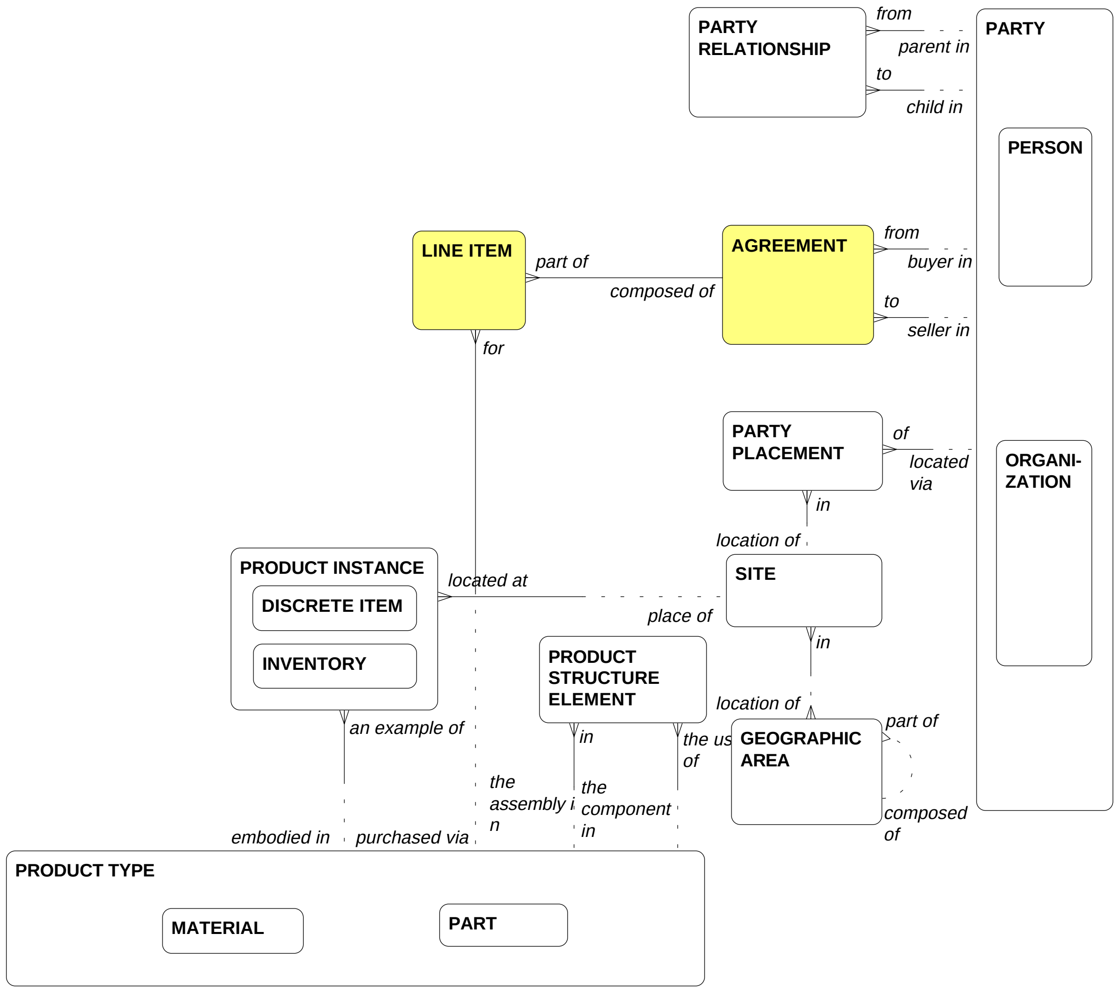

Figure 3: Agreements

Activities are the things the organization does to carry out its business. This is shown in Figure 4. As with PRODUCT TYPES and PRODUCT INSTANCES, there is a distinction drawn between ACTIVITY TYPES (the definition of what is to be done) and ACTIVITIES (the actual doing of it). Attributes of an ACTIVITY TYPE include its description and a standard length of time it is expected to require, while attributes of ACTIVITY include the actual date it occurred and the actual time it took.

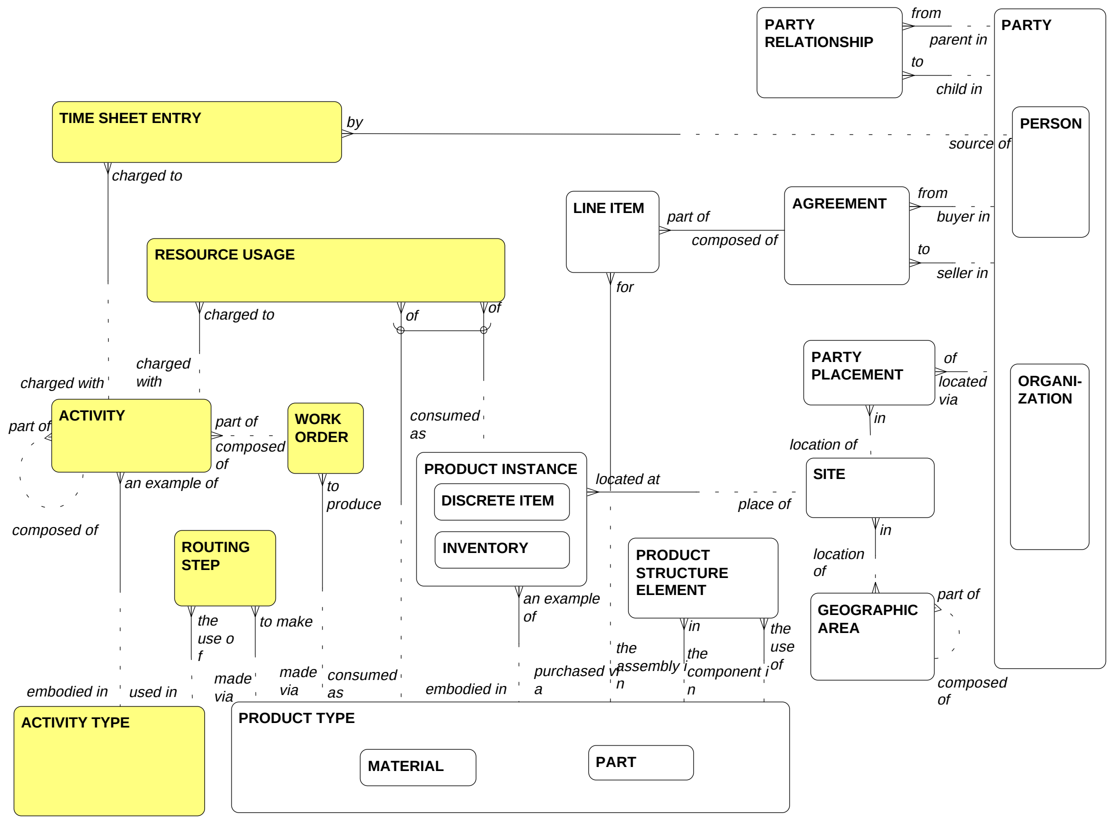

Figure 4: Activities

Parameters

The above model is a good start, but it is not adequate to describe certain common situations. For example, there is a problem with PRODUCT TYPE and PRODUCT INSTANCE. For each of these to be an entity suggests that the attributes for all occurrences of each are the same. This simply is not true.

The attributes of a compressor are quite different from the attributes of a computer or a barrel of crude oil. We would like to have a single concept for "Product", but that concept has many different flavors.

We could define a sub-type for each PRODUCT TYPE, but new product types are being invented all the time, and the data management task would be impossible.

Figure 5 shows PARAMETER itself as an entity. A PARAMETER is a characteristic that is used to define a PRODUCT TYPE.

A PARAMETER ASSIGNMENT is the fact that a particular PARAMETER is used to define a particular PRODUCT TYPE. For example, the PARAMETER "capacity" might be used to describe a boiler, while the PARAMETER "interest rate" might be used to define a savings account.

(Yes, one of the advantages of this approach is that it works as well for banks as it does for nuclear power plants.)

Note that the PARAMETER may be *expressed in* a UNIT OF MEASURE. That boiler "capacity" for example, might be in "cubic feet". The UNIT OF MEASURE that is the *term for* a PARAMETER ASSIGNMENT can override the default UNIT OF MEASURE of the PARAMETER by itself.

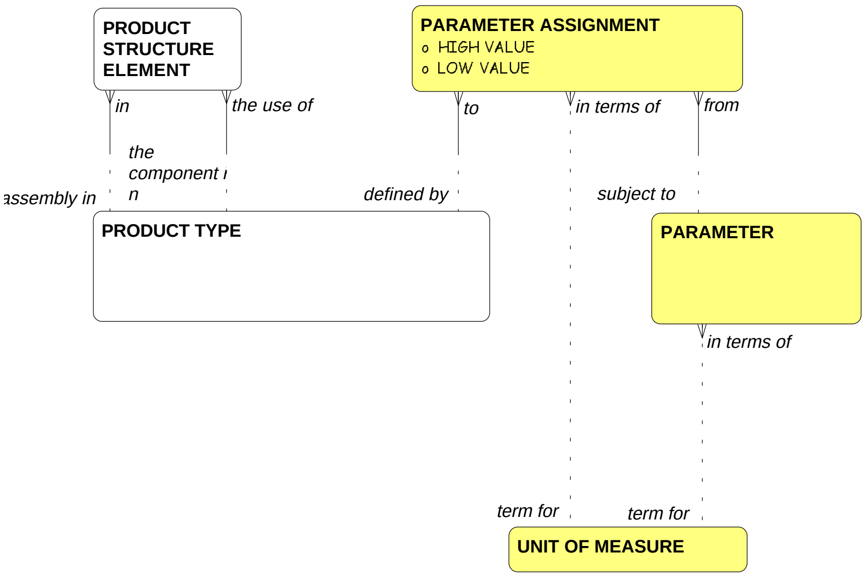

Figure 5: Parameter Assignments

Note that in Figure 6 three kinds of PARAMETERS are shown: A DISCRETE LIST is a parameter that can take only one of a specified set of ALLOWABLE VALUES. For example a "pharmacological category" for a pharmaceutical would have a DISCRETE LIST of ALLOWABLE VALUES. A DERIVED PARAMETER is calculated from one or more other PARAMETERS and/or constants. Each DERIVED PARAMETER must be *calculated from* one or more PARAMETER DERIVATIONS, where each PARAMETER DERIVATION represents a formula of some kind. The formula, in turn, must be *composed of* one or more PARAMETER DERVATION ELEMENTS, where each PARAMETER DERIVATION ELEMENT may be *the use of* another or the use of a constant.

OTHER PARAMETERS simply describe the PRODUCT TYPE. If numeric, these cold be constrained by a "high value" and a "low value". Within these constraints, a PARAMETER ASSIGNMENT could have its own "high value" and "low value".

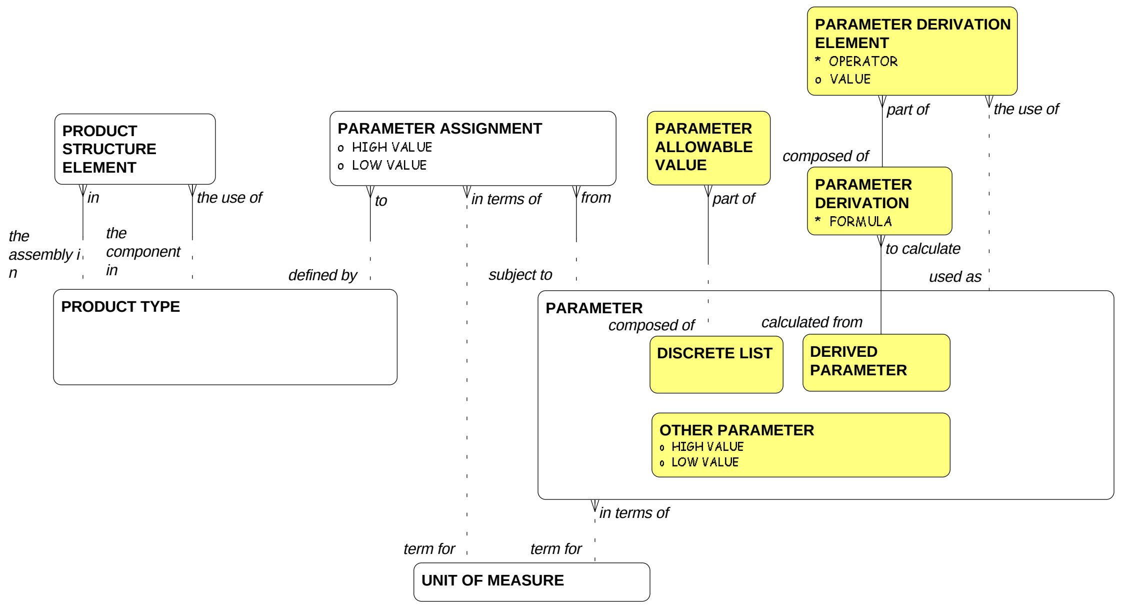

Figure 6: Parameters

A set of PARAMETER ASSIGNMENTS defines the nature of a PRODUCT TYPE. Any PRODUCT INSTANCE that is *an example of* the PRODUCT TYPE is then evaluated with values for the PARAMETERS assigned to its associated PRODUCT TYPE.

Figure 7 shows this. Here a PARAMETER VALUE is the fact that a particular PRODUCT INSTANCE takes a specified "value" *of* a PARAMETER. Note that the arc here is less about the fact that some are *of* a PARAMETER and some are *of* a PARAMETER ASSIGNMENT, than it is about the fact that you can model it either way. If you specify that the value is of a PARAMETER ASSIGNMENT you are keeping PARAMETERS from being specified that were not previously assigned to PRODUCT TYPES. This is a partial business rule, although it still does not require (as a business rule should) that the PRODUCT TYPE and PARAMETER that the PARAMETER VALUE is for represent a legal combination as expressed by PARAMETER ASSIGNMENTS.

If the product type "Model 770 ThinkPad™", for example, had assigned to it the PARAMETER "processor speed", the corresponding PARAMETER VALUE for the particular one I am looking at could be "233" (UNIT OF MEASURE: mhz).

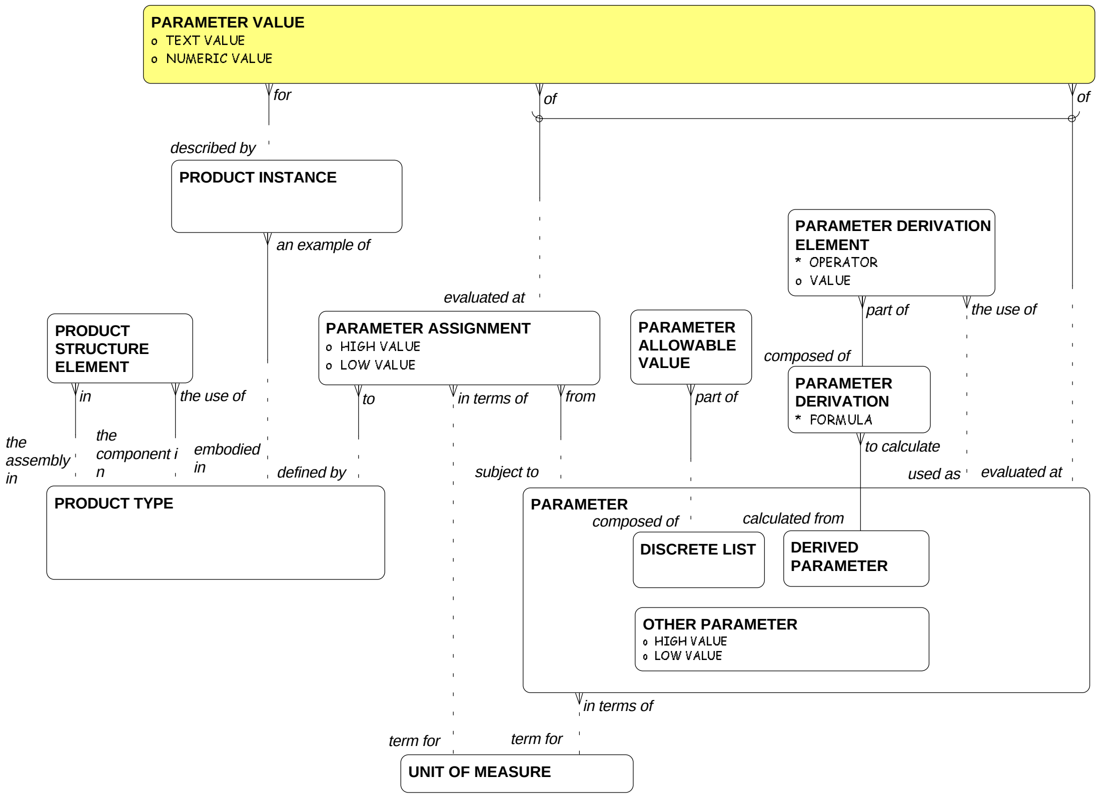

Figure 7: Parameter Values

THE LABORATORY

Your author discovered this structure when doing work for a bank. Sometime thereafter he was working for a lumber products company that needed a model for its laboratory. Fortunately he had been doing the bank work, so he was fully prepared, coming with the following variation:

The laboratory does tests on product samples. In this case (unlike others I came across later), the company knows the PRODUCT TYPE it is dealing with. The tests are simply to determine specific characteristics of the product. For this reason, it is possible to ascertain what the expected characteristics are to be.

In Figure 8 it can be seen that each PRODUCT TYPE may be *evaluated in terms of* one or more EXPECTED OBSERVATIONS each of which is *of* a particular VARIABLE. That is, the PRODUCT TYPE is considered to be within specifications if the value of a VARIABLE is between a "high value" and "low value" specified in the EXPECTED OBSERVATION.

The laboratory process begins with a SAMPLE being taken from a PRODUCT INSTANCE (which is *an example of* the

PRODUCT TYPE in question). This sample is then *subject to* one or more LABORATORY TESTS. Each LABORATORY TEST, in turn, is *the source of* one or more OBSERVATIONS – each *on* a VARIABLE.

If you rename VARIABLE to PARAMETER, EXPECTED OBSERVATION to PARAMETER ASSIGNMENT, and OBSERVATION to PARAMETER VALUE, and if you then collapse SAMPLE and LABORATORY TEST, you have the model shown above in Figure 7.

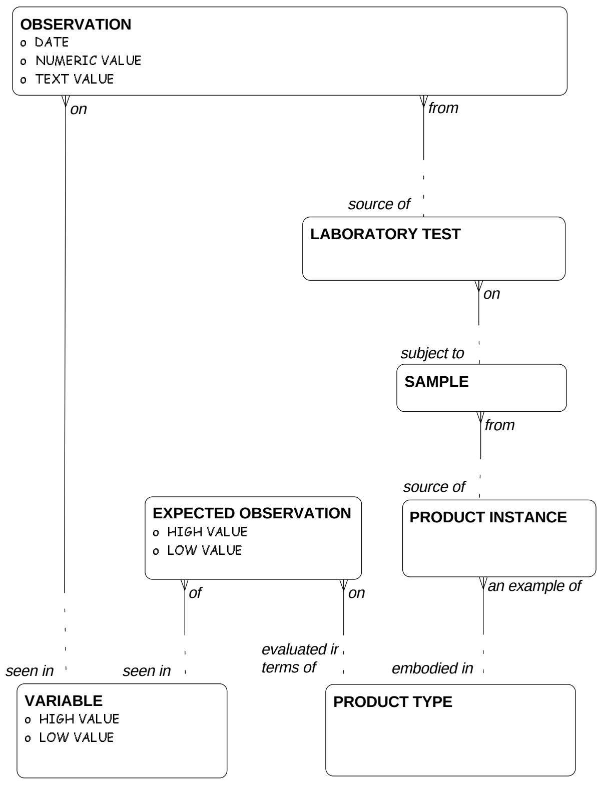

Figure 8 : The Laboratory

CLINICAL RESEARCH

This parameterization idea got stretched even further when applied to the collection of data from clinical pharmaceutical trials.

Pharmaceutical research is an example of a particularly messy modeling problem: Clinical data are captured on “case report forms” (CRFs), which, which depending on the study – indeed, depending on the part of the study – have a variable number of sections, where each section could have one or several numbers, pieces of text, or even drawings. There is no fundamental, underlying structure here. The only way to address the problem is to go up one level of abstraction.

Figure 9 shows how a clinical STUDY is defined to be *composed of* one or more VISIT SPECIFICATIONS by a patient to a physician. Each visit has been planned in the design of the study, as to what information is to be collected. This information is organized into STANDARD BLOCKS, such as "personal information", "hematological information", "cardio-vascular information", and so forth. Each STANDARD BLOCK is defined in terms of the BLOCK VARIABLES it is *composed of*, where a BLOCK VARIABLE is *the use of* a VARIABLE as *part of* a STANDARD BLOCK.

The standards are defined by the pharmaceutical company, but each block may be tailored (to some extent) to the study in a VISIT BLOCK, which is *part of* a VISIT SPECIFICATION. Each VISIT BLOCK then may have its own definitions of which VISIT BLOCK VARIABLES are *part of* it. (A business rule defined by the research company determines the extent to which VISIT BLOCKS must conform to the specifications of a STANDARD BLOCK.)

Once the CRFs have been defined as to what VISIT BLOCKS and VISIT BLOCK VARIABLES each VISIT SPECIFICATION contains, data may be collected. Each element on the CRF is an OBSERVATION, which may be either text or numeric, and which is collected at a specific date *from* a VISIT *by* a specific PATIENT.

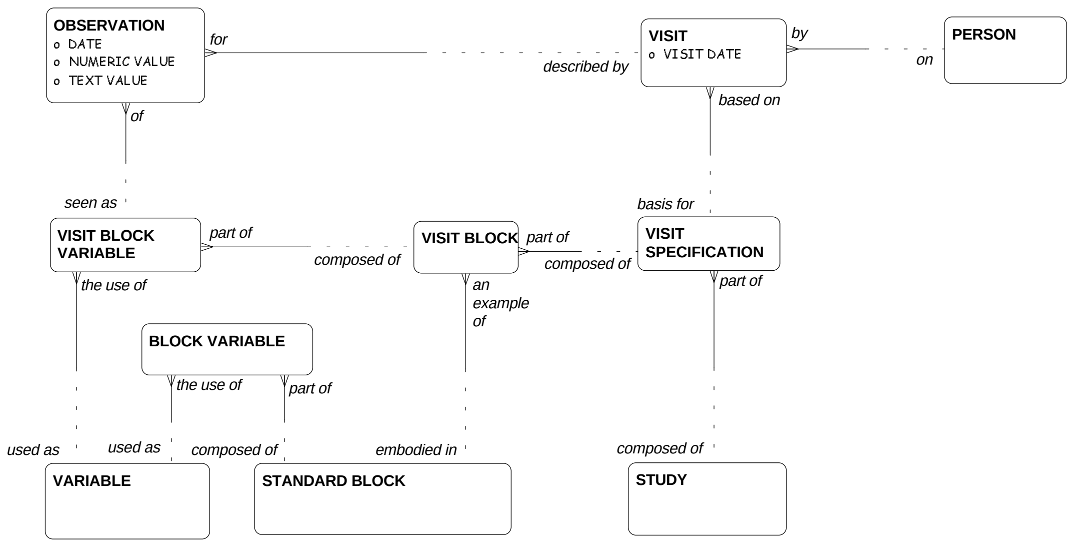

Figure 9: Clinical Trials

It may be argued that, while this is the most orderly way to capture all these data, it makes them a little difficult to get at. To correlate measurements of two variables it is necessary to construct a query that asks for all values of a particular variable and the circumstances of their collection, in conjunction with all values of another variable, where the circumstances of their collection are matched with the circumstances of the first. This is hard.

To address this, the pharmaceutical companies that have taken this approach have devised a table structure derived from this one. (This was the original "data mart" before that word became fashionable.) The idea is that what the statisticians want to see is all the data of a certain kind together.

It turns out that the "block" structure described above gives us the opportunity to "de-abstract" the data into something a little more manageable. It is possible to write a single utility program that takes the OBSERVATION data and reorganize it into a single table for each VISIT BLOCK, with the VARIABLES showing up as columns in this table.

This appears in Figure 10. Each table represents a VISIT BLOCK, and the columns allow statistical analysis of correlations between similar variables. Even correlations between variables in different tables is easier that it was in the original OBSERVATIONS table.

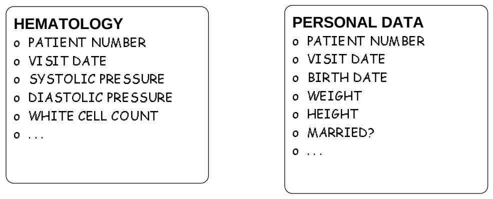

Figure 10: "De-abstracted" Clinical Data

MAPPING LEGACY SYSTEMS

Data modeling is not done in a vacuum. It's often done in conjunction with a major project. These days, that project is as likely as not to build a "data warehouse" – a repository that is supposed to hold all of a company's data and make them available to management for inspection and analysis.

The problem with building a data warehouse is that, while a data model is valuable in defining it's architecture, it doesn't help much in dealing with all those old "legacy" systems that are going to be the source of the data. The designers of those systems often were not very cooperative in clearly identifying exactly what each datum means and where it fits into the larger scheme of things.

The data model does help, in that it provides a road map of what kind of data have to be in there somewhere. What is needed next, though, is some sort of mapping from the columns and tables (fields and files?) of the old systems to the attributes and entities of the model.

In one sense, this is not a logical data modeling problem. After all, the legacy database designs are physical structures, not logical ones. The assignment, however, is to make these logical structures useful, and it is our job to do so.

So, it is necessary to look at the model of our "metadata repository" that is keeping the "model of our models". In Figure 11, you can see this mapping. The legacy system consists (for the sake of argument – we will not get into more complex legacy systems) of TABLES, each of which is *composed of* one or more COLUMNS. Our model, on the other hand, is made up of ENTITIES, each of which is *composed of* one or more ATTRIBUTES. If life were simple, all we would have to do is to create a COLUMN MAPPING *of* each COLUMN of the legacy system *to* an ATTRIBUTE which is *part of* an ENTITY in our model.

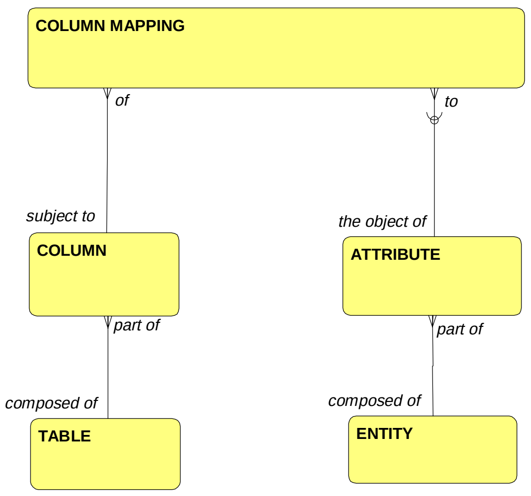

Figure 11: Simple Mapping

Alas, life is not so simple. If you remember, much of the data described in our model is not contained in attributes of the entities, but separately in PARAMETER VALUES. That is, the definition of the data structure is not in the entities and attributes at all, but in PARAMETER ASSIGNMENTS. This means that in some cases, we are not mapping a COLUMN to an ATTRIBUTE, but to a PARAMETER ASSIGNMENT, as shown in Figure 12. Here, the COLUMN MAPPING is either *to* an ATTRIBUTE or *to* a PARAMETER ASSIGNMENT. Note that for the mapping to a PARAMETER ASSIGN-MENT to work, the TABLE involved had better have something to do with PRODUCT TYPES.

In this view of the world, note that the PARAMETER VALUES *of* a PARAMETER are exactly like the values *for* an ATTRIBUTE. As implemented in a relational database, of course, those values will go *into* a column in the table corresponding to the entity, rather than being captured in a separate table, but conceptually, a value of an attribute is exactly equivalent to a value of a parameter.

Note that we have brought together the application model of our business with the meta-model that is supposed to define the application model. Stay tuned. This gets weirder.

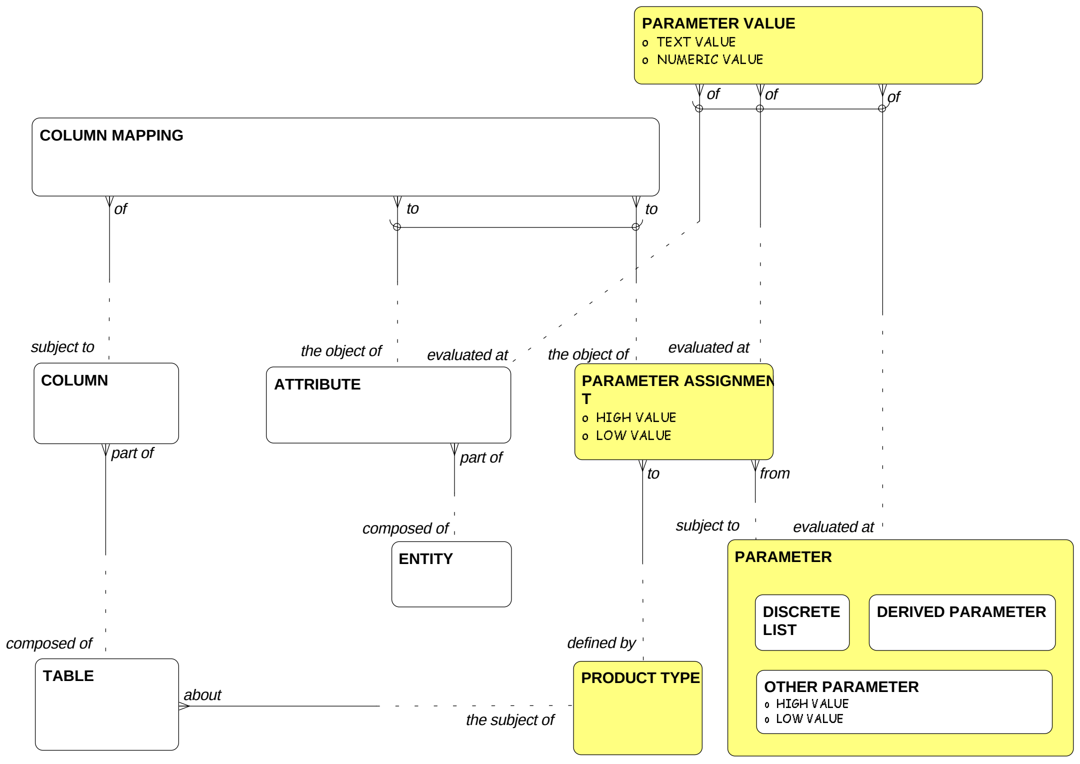

Figure 12: Not So Simple Mapping

Let's look at our model of entities and attributes. (See Figure 13.)[2]

You will recall that each ENTITY is *composed of* one or more attributes. Looking more closely at ATTRIBUTES, we can see that each ATTRIBUTE must be *constrained by* a DOMAIN. The DOMAIN provides validation rules for the ATTRIBUTE. Looking at DOMAINS carefully, it turns out that there are at least three kinds. There are OTHER DOMAINS, that simply provide a format

[2] For the attribute model and the consolidated Attribute Meta Model which follows, your author is indebted to *Allan Kolber*, of Butler Technology Solutions, Inc.

and perhaps limits to numeric values; VALUE SETS, which require the attribute to take one of a specified set of ATTRIBUTE ALLOWABLE VALUES; and CALCULATIONS, which are derived from other domains.

Now, look carefully. Does this model look familiar? We just did it in the PARAMETER exercise. Figure 14 brings all this together. The model on the right, from our application model, *is* the model of entities and attributes from the metadata repository. An ATTRIBUTE is nothing but a PARAMETER ASSIGNMENT in disguise. A DOMAIN is a PARAMETER.

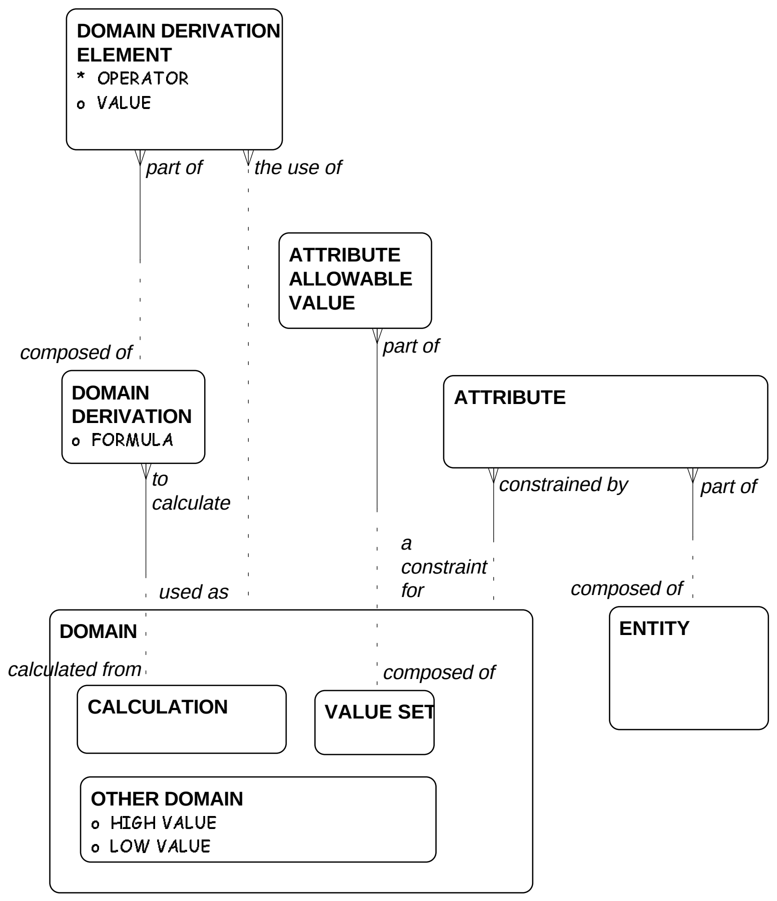

Figure 13: Entities and Attributes

Now, what about that ENTITY on the left and the PRODUCT TYPE on the right? Well, it so happens that the PARAMETER model we've shown here is but a specialized example of a more general phenomenon.

There are different kinds of PARTIES, for example, and we want to collect different kinds of parameters for them: departments, individuals, housholds, professional societies, etc. Similarly, different TRANSACTION TYPES might be *defined by* different PARAMETER ASSIGNMENTS. In general, there are few enough of these that we can use the super-type/sub-type structure – you know, the device we use when there are different sub-categories of things that have different attributes? When we use sub-types, conceptually, we are doing exactly the same thing we do here with PARAMETER ASSIGNMENTS.

Each PARAMETER ASSIGNMENT could as easily be *to* a PRODUCT TYPE, a PARTY TYPE, a TRANSACTION TYPE, or anything else. That is, it could be *to* any ENTITY.

All of this is to say that pretty much everything in the world (with the possible exception of accounting) can be represented by the "universal data model", shown in Figure 15.

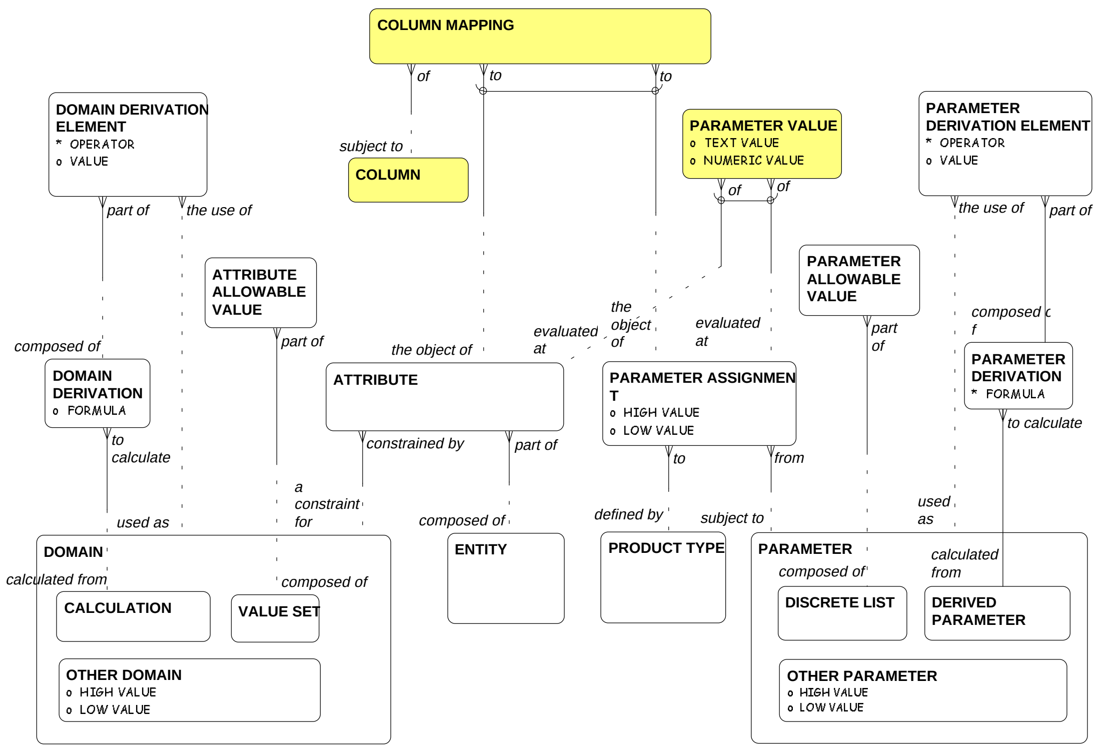

Figure 14: Attribute Meta Model

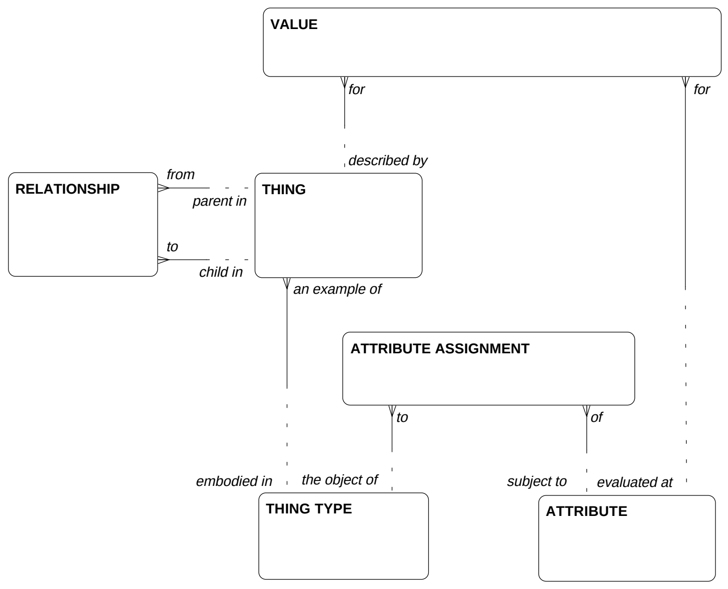

Figure 15: The Universal Data Model

ABOUT THE AUTHOR

A twenty-five year veteran of the Information Industry, with extensive experience developing on-line, database-oriented systems, Dave Hay has been producing data models to support strategic information planning and requirements analysis for ten years. He has worked in a variety of industries, including, among others, power generation, clinical pharmaceutical research, cable television, oil refining, and forestry.

This diversity of industrial experience prompted him to write *Data Model Patterns: Conventions of Thought*, published by Dorset House Publishers, which presents data models of common business situations that cross industry.

Dave is President of Essential Strategies, Inc., a consulting firm dedicated to helping clients define corporate information architecture, identify requirements, build, and implement new systems.

You can read more of his works on the Essential Strategies, Inc. web page, *http://www.essentialstrategies.com*.